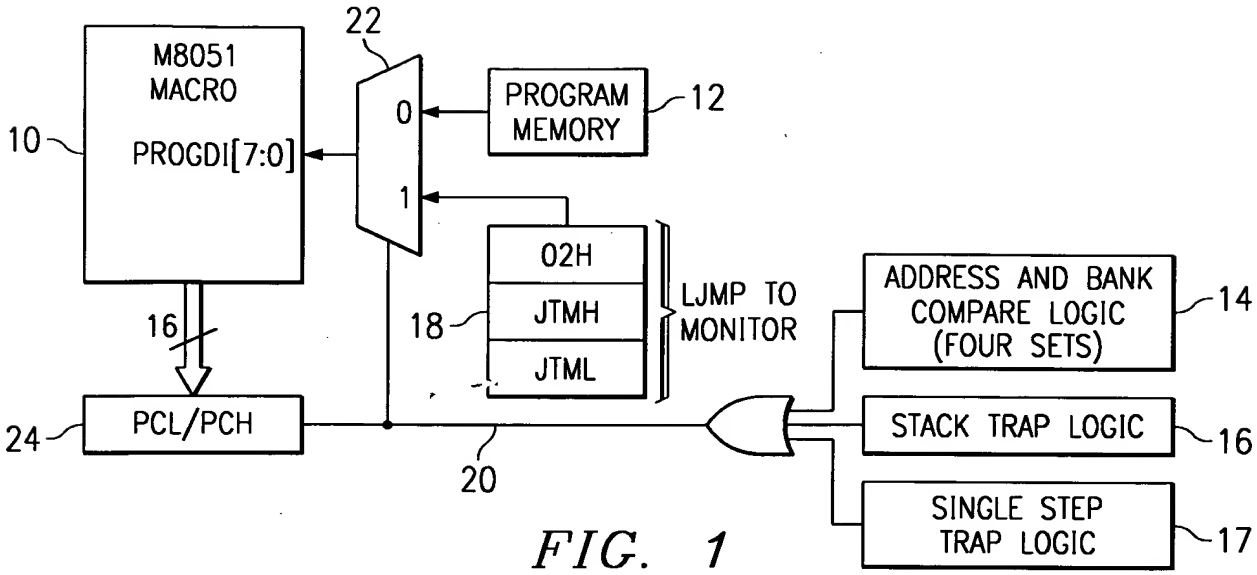
M8051
MACRO
PROGDI[7:0]
10
22
0
1
PROGRAM
MEMORY
12
02H
JTMH
JTML
18
LJMP TO
MONITOR
16
PCL/PCH
24
20
ADDRESS AND BANK
COMPARE LOGIC
(FOUR SETS)
14
STACK TRAP LOGIC
16
SINGLE STEP
TRAP LOGIC
17

FIG. 1

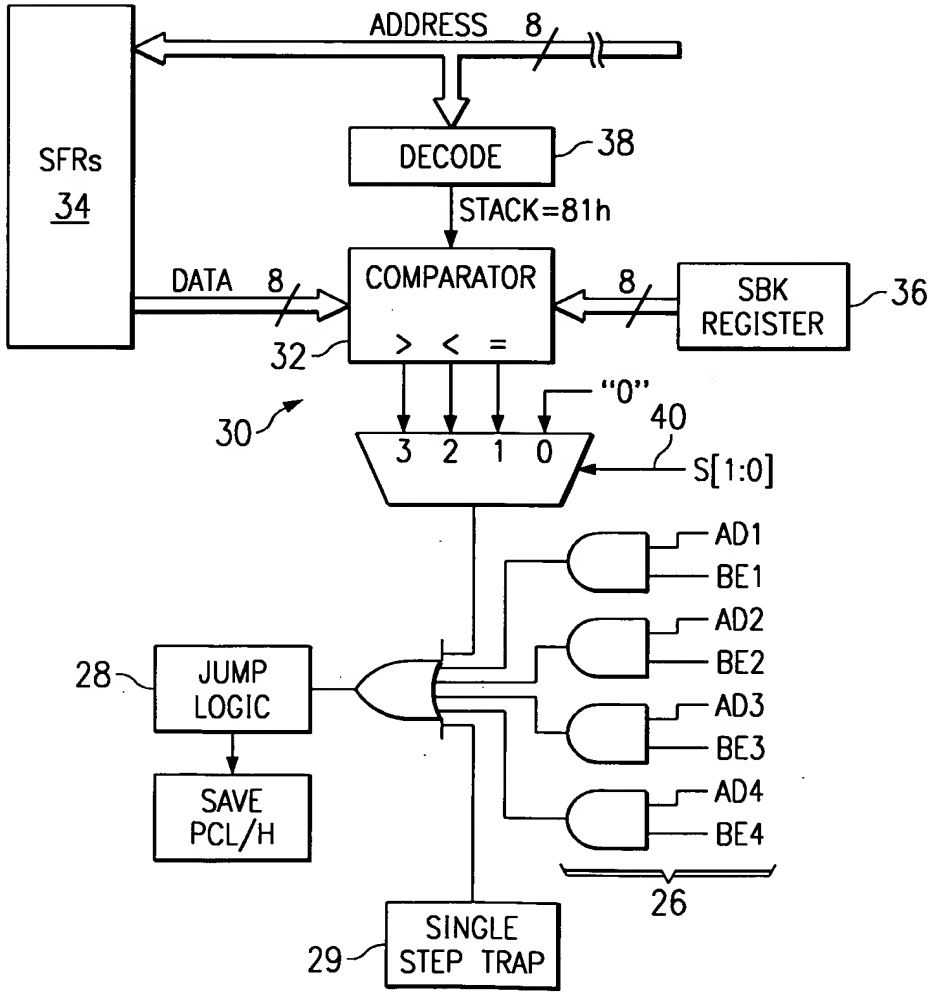
SFRs
34
ADDRESS 8
DECODE
38
STACK=81h
DATA 8
COMPARATOR
> < =
32
8
SBK
REGISTER
36
30
"0"
40
3 2 1 0
S[1:0]
AD1
BE1
AD2
BE2
AD3
BE3
AD4
BE4
26
28
JUMP
LOGIC
SAVE
PCL/H
29
SINGLE
STEP TRAP

FIG. 2

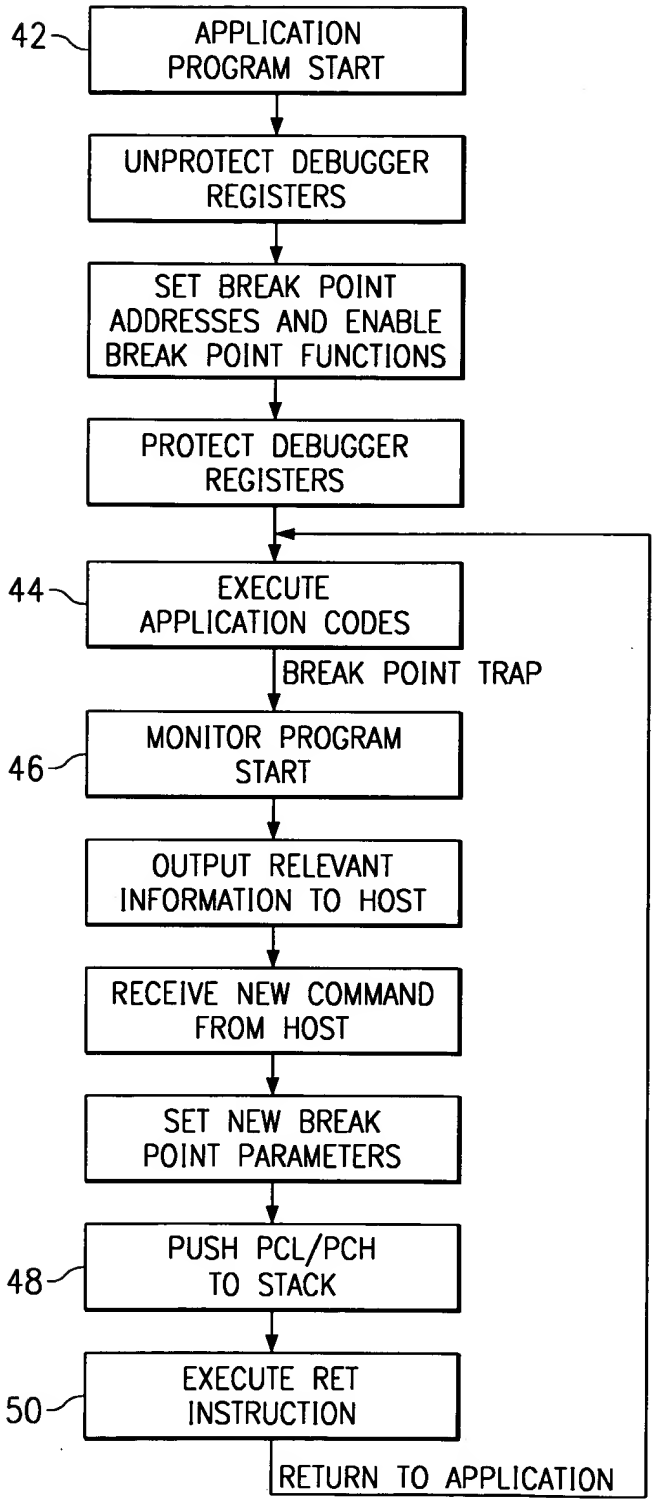
42
APPLICATION PROGRAM START
UNPROTECT DEBUGGER REGISTERS
SET BREAK POINT ADDRESSES AND ENABLE BREAK POINT FUNCTIONS
PROTECT DEBUGGER REGISTERS
44
EXECUTE APPLICATION CODES
BREAK POINT TRAP
46
MONITOR PROGRAM START
OUTPUT RELEVANT INFORMATION TO HOST
RECEIVE NEW COMMAND FROM HOST
SET NEW BREAK POINT PARAMETERS
48
PUSH PCL/PCH TO STACK
50
EXECUTE RET INSTRUCTION
RETURN TO APPLICATION

FIG. 3

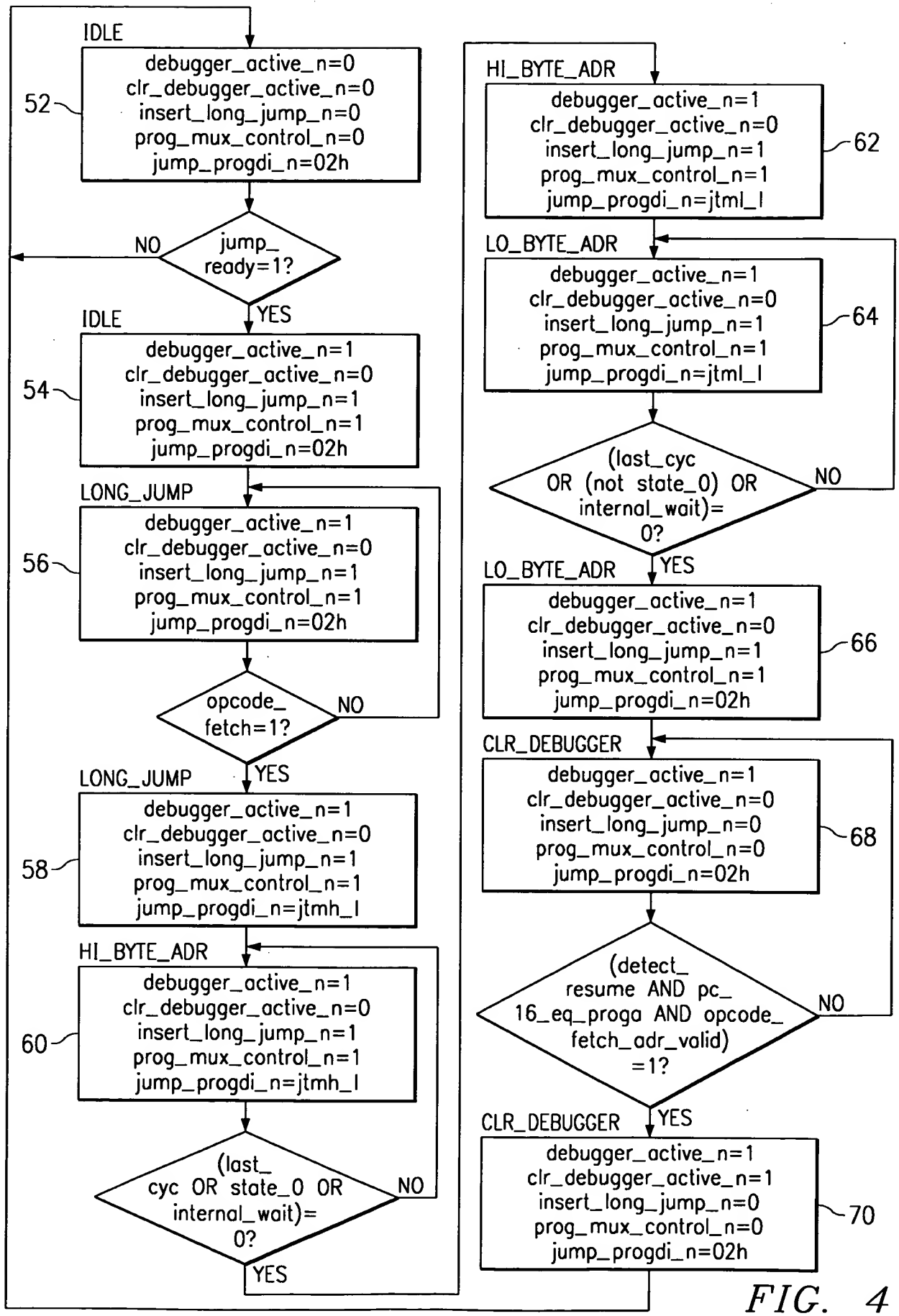
IDLE
debugger_active_n=0
clr_debugger_active_n=0
insert_long_jump_n=0
prog_mux_control_n=0
jump_progdi_n=02h
52
jump_ready=1?
NO
YES
IDLE
debugger_active_n=1
clr_debugger_active_n=0
insert_long_jump_n=1
prog_mux_control_n=1
jump_progdi_n=02h
54
LONG_JUMP
debugger_active_n=1
clr_debugger_active_n=0
insert_long_jump_n=1
prog_mux_control_n=1
jump_progdi_n=02h
56
opcode_fetch=1?
NO
YES
LONG_JUMP
debugger_active_n=1
clr_debugger_active_n=0
insert_long_jump_n=1
prog_mux_control_n=1
jump_progdi_n=jtmh_l
58
HI_BYTE_ADR
debugger_active_n=1
clr_debugger_active_n=0
insert_long_jump_n=1
prog_mux_control_n=1
jump_progdi_n=jtmh_l
60
(last_cyc OR state_0 OR internal_wait)=0?
NO
YES
HI_BYTE_ADR
debugger_active_n=1
clr_debugger_active_n=0
insert_long_jump_n=1
prog_mux_control_n=1
jump_progdi_n=jtml_l
62
LO_BYTE_ADR
debugger_active_n=1
clr_debugger_active_n=0
insert_long_jump_n=1
prog_mux_control_n=1
jump_progdi_n=jtml_l
64
(last_cyc OR (not state_0) OR internal_wait)=0?
NO
YES
LO_BYTE_ADR
debugger_active_n=1
clr_debugger_active_n=0
insert_long_jump_n=1
prog_mux_control_n=1
jump_progdi_n=02h
66
CLR_DEBUGGER
debugger_active_n=1
clr_debugger_active_n=0
insert_long_jump_n=0
prog_mux_control_n=0
jump_progdi_n=02h
68
(detect_resume AND pc_16_eq_proga AND opcode_fetch_adr_valid)=1?
NO
YES
CLR_DEBUGGER
debugger_active_n=1
clr_debugger_active_n=1
insert_long_jump_n=0
prog_mux_control_n=0
jump_progdi_n=02h
70

FIG. 4

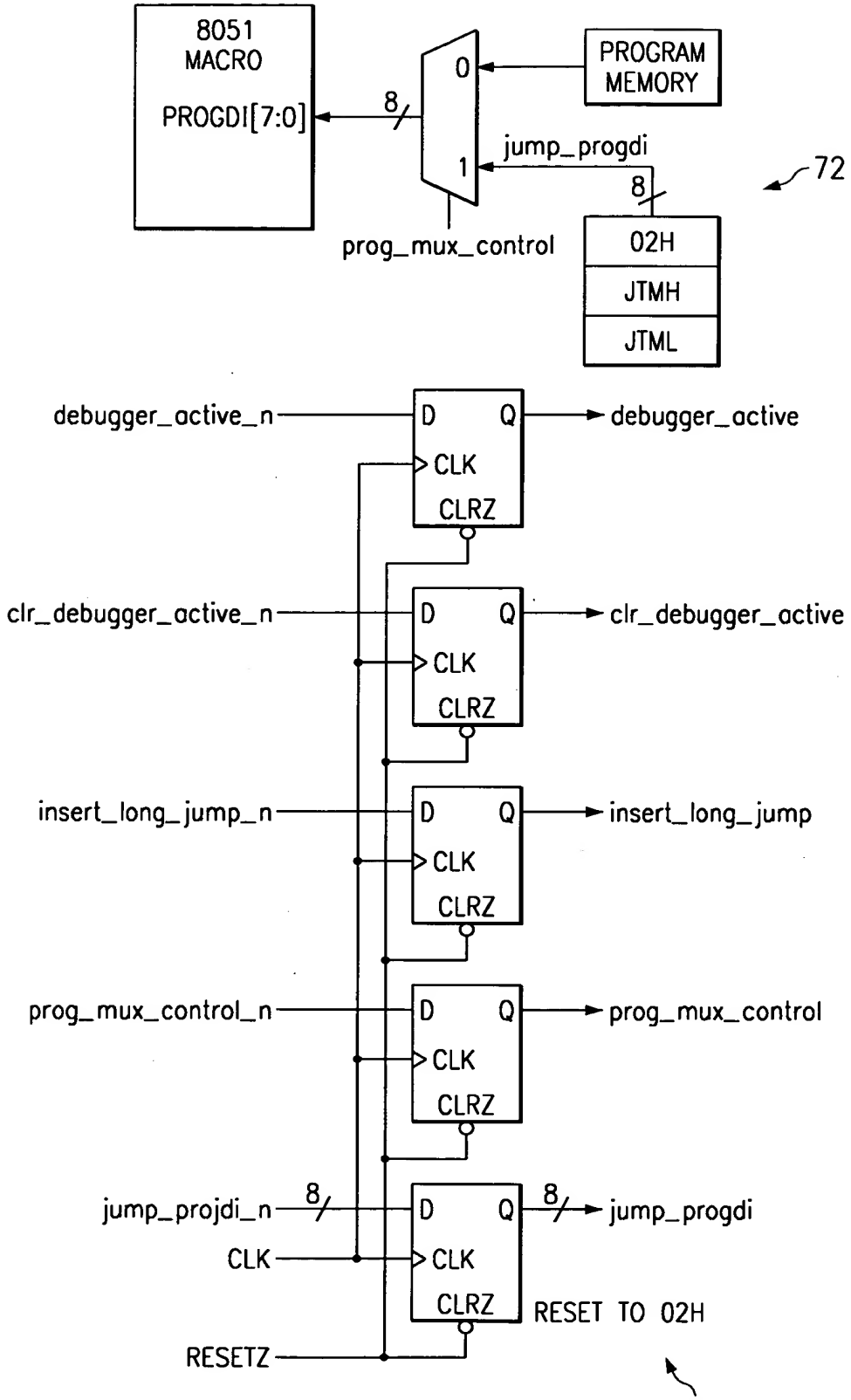
8051
MACRO
PROGDI[7:0]
8
0
1
PROGRAM
MEMORY
jump_progdi
8
72
prog_mux_control
02H
JTMH
JTML
debugger_active_n
D
Q
CLK
CLRZ
debugger_active
clr_debugger_active_n
clr_debugger_active
insert_long_jump_n
insert_long_jump
prog_mux_control_n
prog_mux_control
jump_projdi_n
8
jump_progdi
CLK
RESET TO 02H
RESETZ
74

FIG. 5

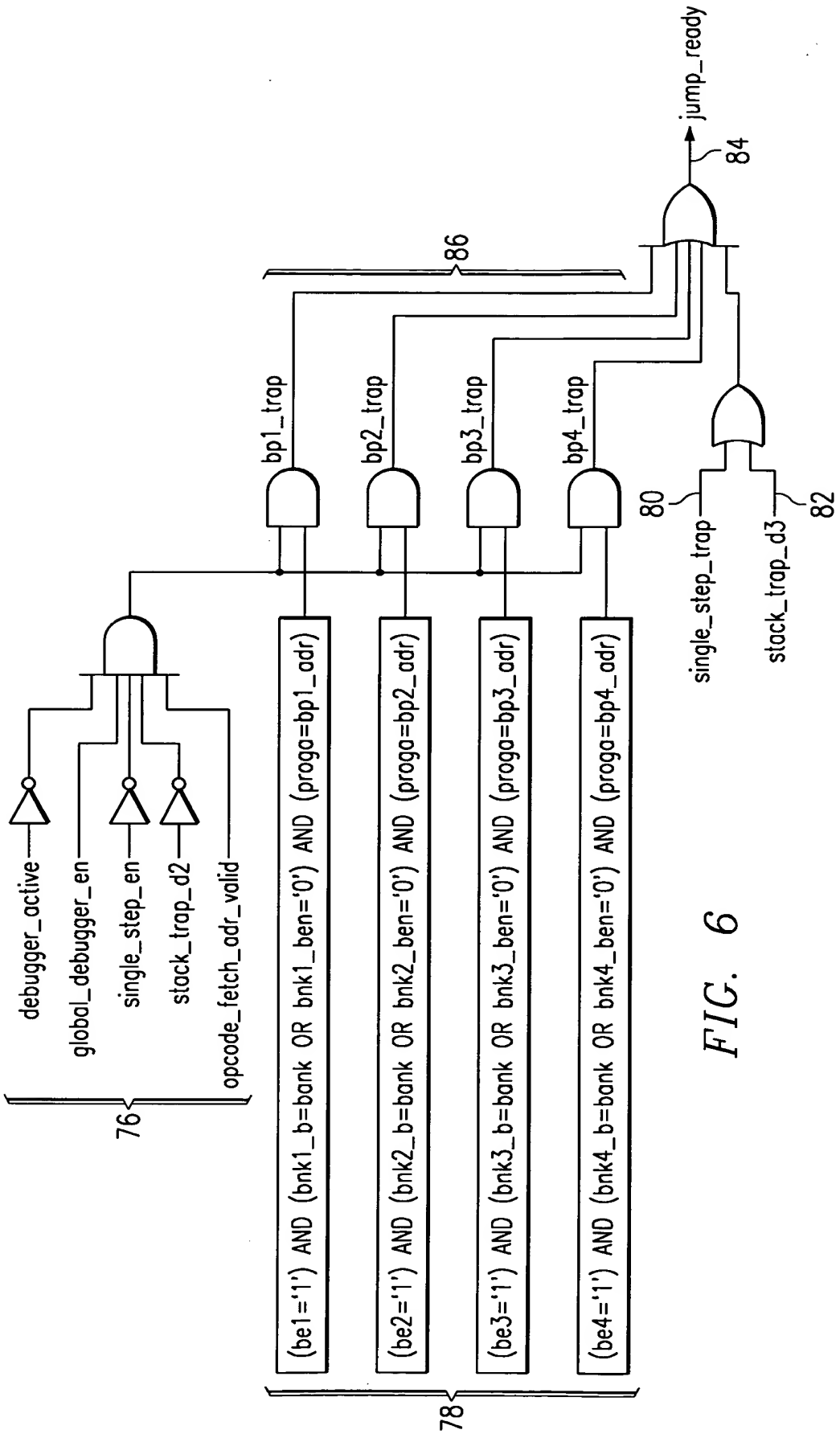
jump_ready
84
86
bp1_trap
bp2_trap
bp3_trap
bp4_trap
80
82
single_step_trap
stack_trap_d3
debugger_active
global_debugger_en
single_step_en
stack_trap_d2
opcode_fetch_adr_valid
76
(be1='1') AND (bnk1_b=bank OR bnk1_ben='0') AND (proga=bp1_adr)
(be2='1') AND (bnk2_b=bank OR bnk2_ben='0') AND (proga=bp2_adr)
(be3='1') AND (bnk3_b=bank OR bnk3_ben='0') AND (proga=bp3_adr)
(be4='1') AND (bnk4_b=bank OR bnk4_ben='0') AND (proga=bp4_adr)
78

FIG. 6

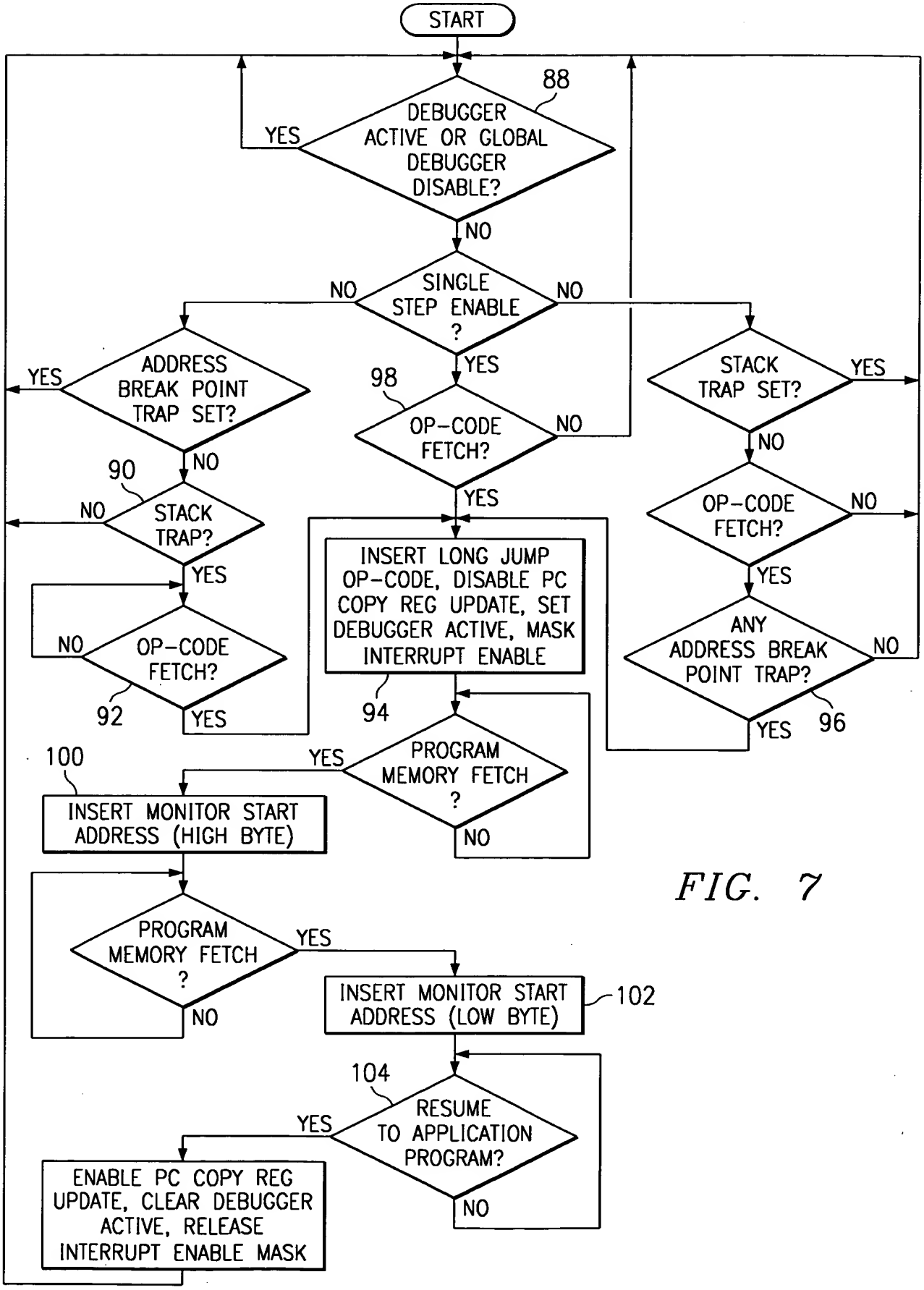
START
88
DEBUGGER ACTIVE OR GLOBAL DEBUGGER DISABLE?
YES
NO
SINGLE STEP ENABLE ?
NO
NO
YES
ADDRESS BREAK POINT TRAP SET?
YES
NO
98
OP-CODE FETCH?
NO
YES
STACK TRAP SET?
YES
NO
90
STACK TRAP?
NO
YES
OP-CODE FETCH?
NO
YES
92
INSERT LONG JUMP OP-CODE, DISABLE PC COPY REG UPDATE, SET DEBUGGER ACTIVE, MASK INTERRUPT ENABLE
94
OP-CODE FETCH?
NO
YES
ANY ADDRESS BREAK POINT TRAP?
NO
YES
96
100
PROGRAM MEMORY FETCH ?
YES
NO
INSERT MONITOR START ADDRESS (HIGH BYTE)
PROGRAM MEMORY FETCH ?
YES
NO
INSERT MONITOR START ADDRESS (LOW BYTE)
102
104
RESUME TO APPLICATION PROGRAM?
YES
NO
ENABLE PC COPY REG UPDATE, CLEAR DEBUGGER ACTIVE, RELEASE INTERRUPT ENABLE MASK

FIG. 7

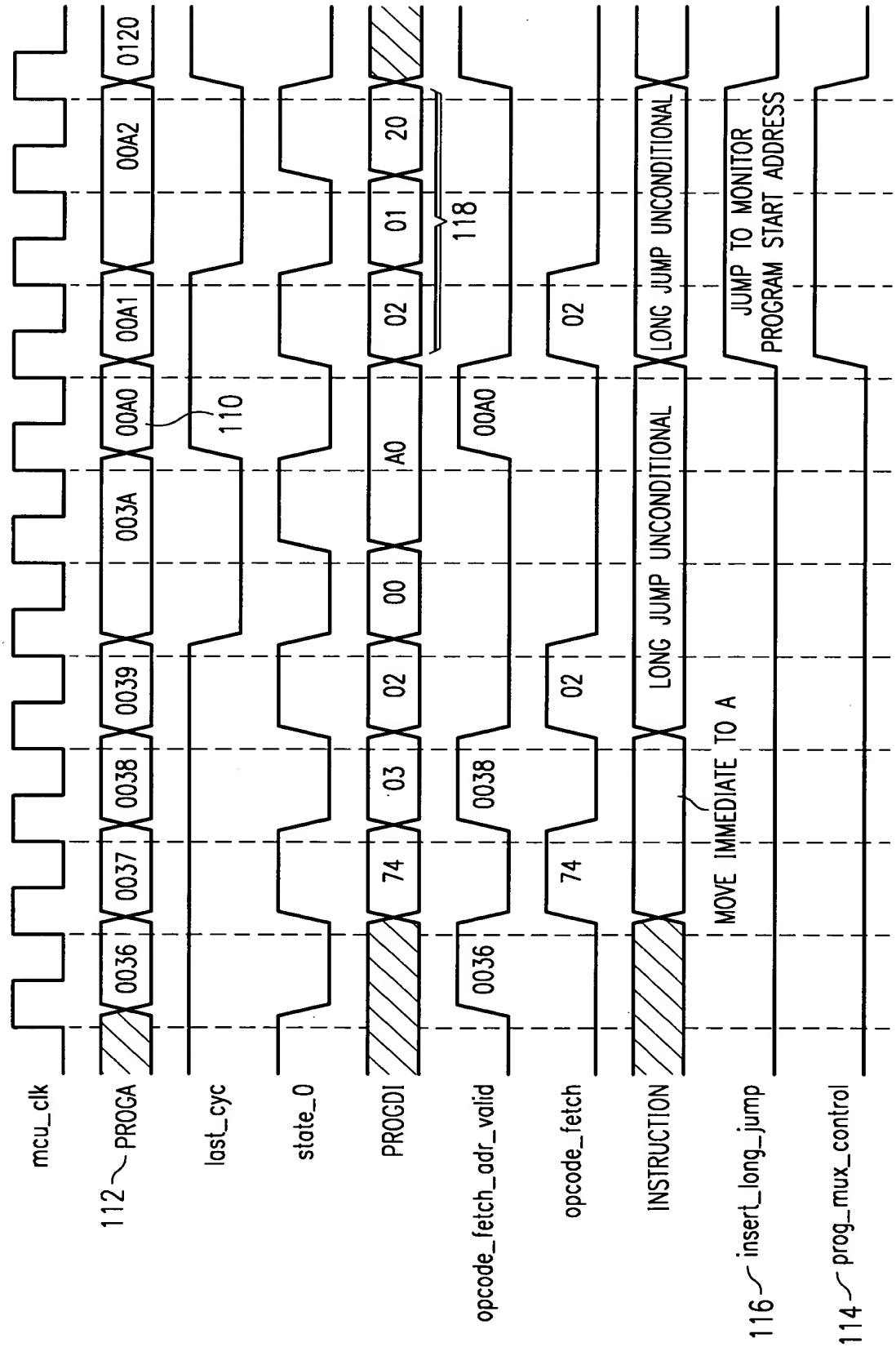
mcu_clk
112 PROGA
last_cyc
state_0
PROGDI
opcode_fetch_adr_valid
opcode_fetch
INSTRUCTION
116 insert_long_jump
114 prog_mux_control
0036
0037
0038
0039
003A
00A0
00A1
00A2
0120
110
74
03
02
00
A0
02
01
20
118
0038
00A0
02
02
MOVE IMMEDIATE TO A
LONG JUMP UNCONDITIONAL
LONG JUMP UNCONDITIONAL
JUMP TO MONITOR
PROGRAM START ADDRESS

FIG. 8a

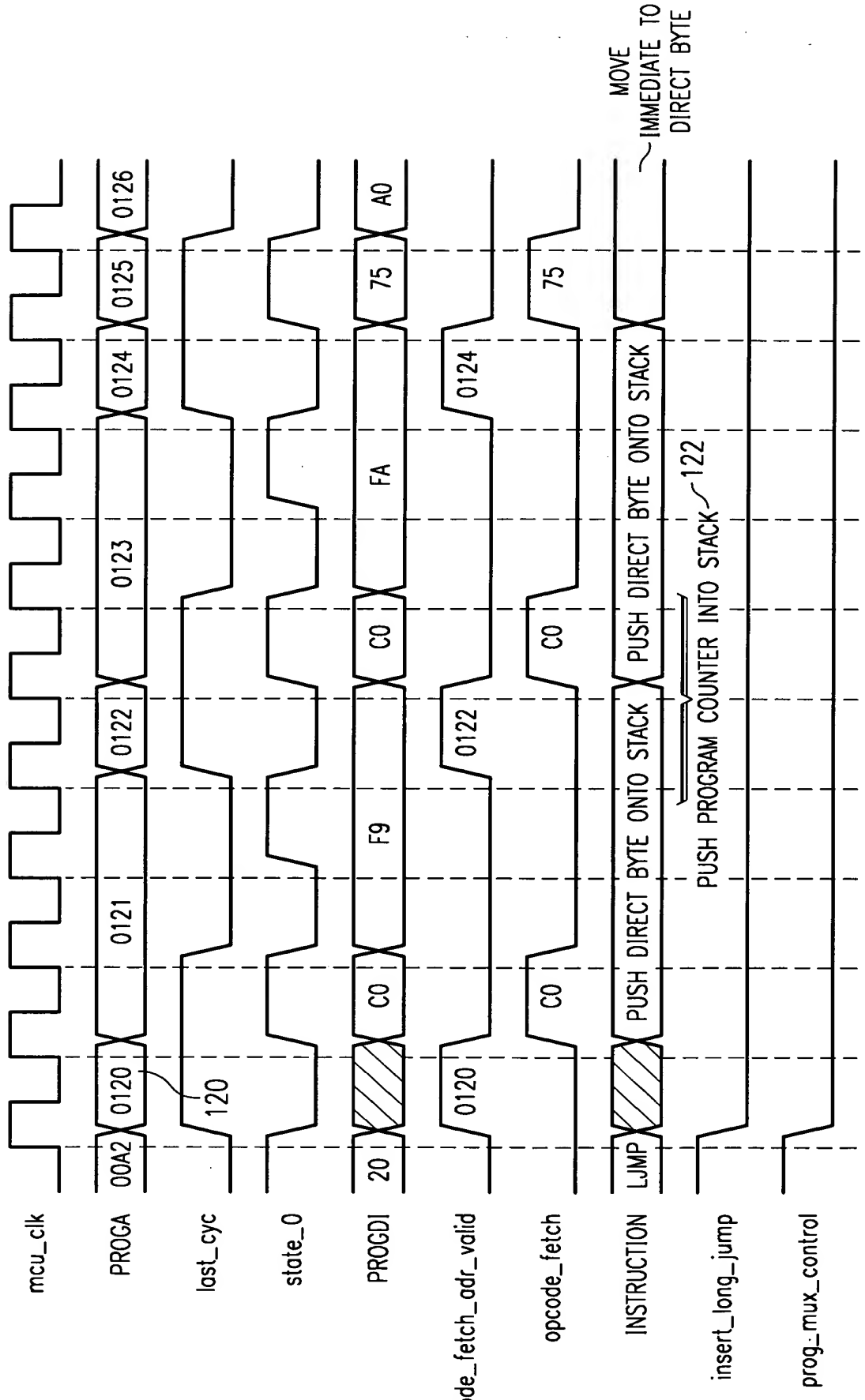
mcu_clk
PROGA
00A2
0120
0121
0122
0123
0124
0125
0126
last_cyc
120
state_0
PROGDI
20
C0
F9
C0
FA
C0
75
A0
opcode_fetch_adr_valid
0120
0122
0124
opcode_fetch
C0
75
INSTRUCTION
LJMP
PUSH DIRECT BYTE ONTO STACK
PUSH DIRECT BYTE ONTO STACK
MOVE IMMEDIATE TO DIRECT BYTE
insert_long_jump
PUSH PROGRAM COUNTER INTO STACK
122
prog_mux_control

FIG. 8b

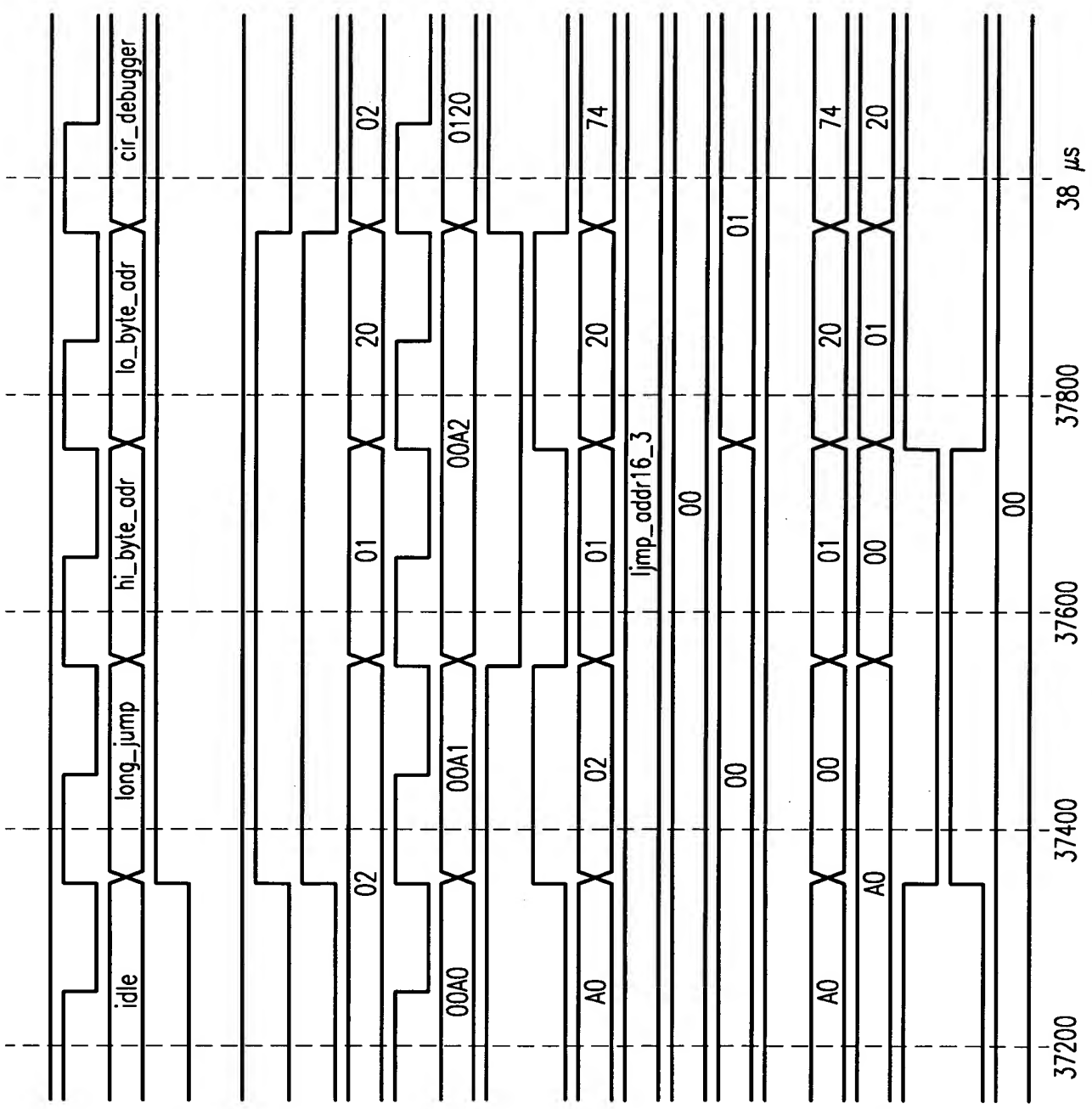
pc_16_eq_progo
clk_i
130 state
debugger_active
clr_debugger_active
insert_long_jump
prog_mux_control
jump_progdi
mcu_clk_i
proga_i
last_cyc_i
state_0_i
progdi
opcode_name
int_mem_data_i
int_mem_wraddress_i
int_mem_wren_i
int_mem_rdaddress_i
rdaddress_reg
int_mem_rden_i
u_int_256ram/rden
int_mem_data_o
idle
long_jump
hi_byte_adr
lo_byte_adr
cir_debugger
02
01
20
02
00A0
00A1
00A2
0120
A0
02
01
20
74
ljmp_addr16_3
00
00
01
A0
00
01
20
74
A0
00
01
20
00
37200
37400
37600
37800
38 µs

FIG. 9a

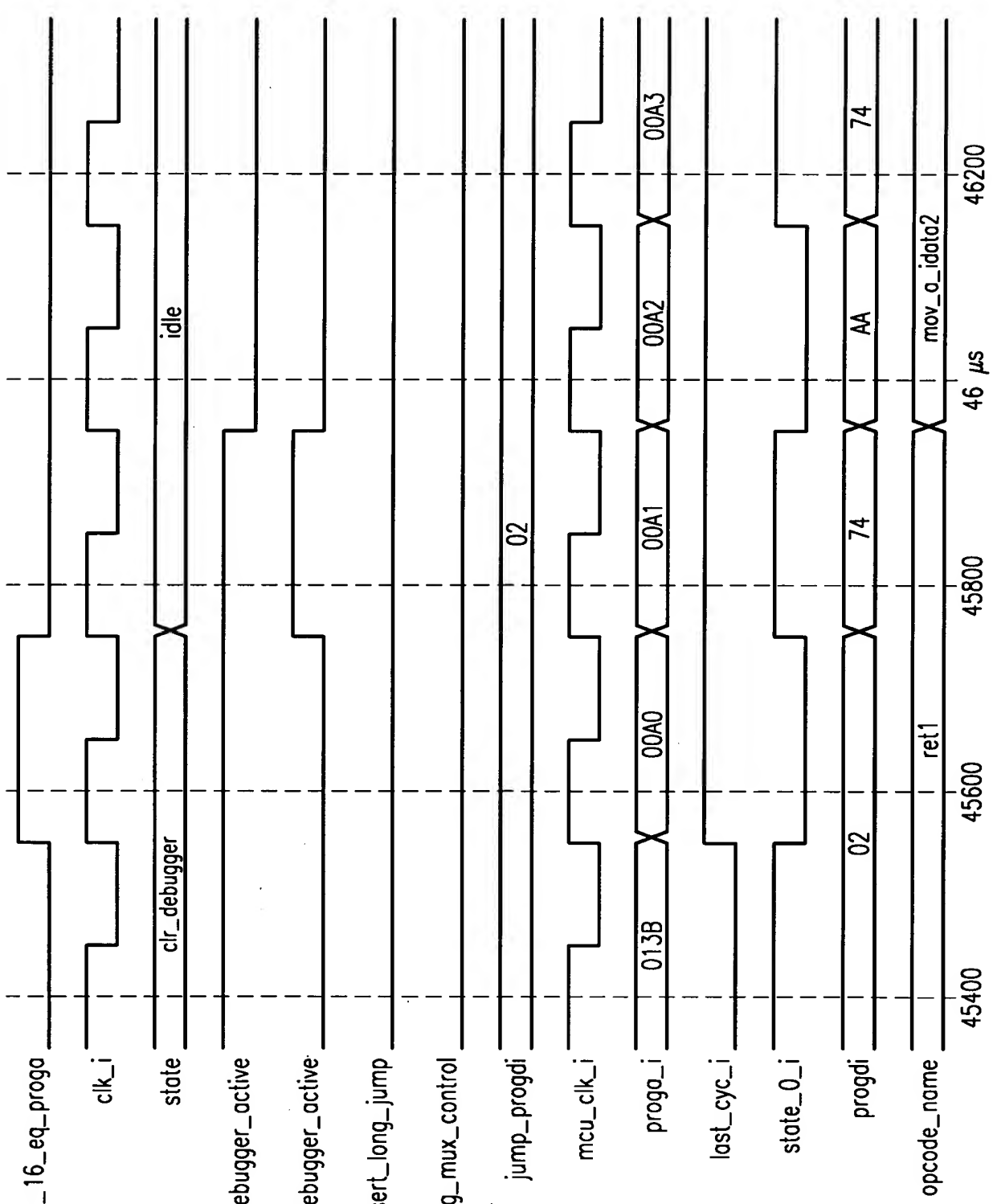
pc_16_eq_proga
clk_i
state
idle
clr_debugger
debugger_active
clr_debugger_active
insert_long_jump
prog_mux_control
jump_progdi
02
mcu_clk_i
proga_i
013B
00A0
00A1
00A2
00A3
last_cyc_i
state_0_i
progdi
02
74
AA
74
opcode_name
ret1
mov_a_idata2
45400
45600
45800
46 µs
46200

FIG. 9b

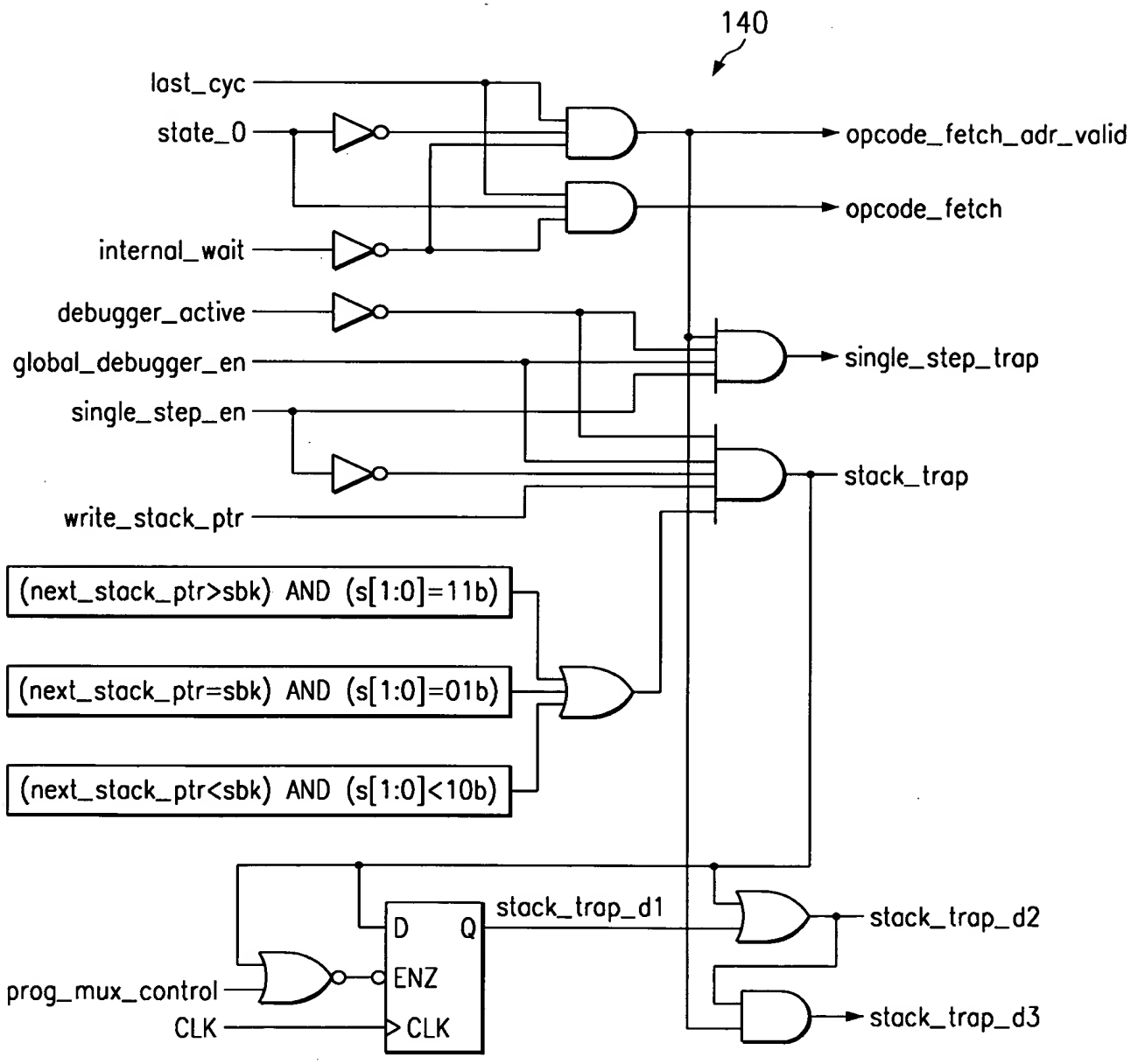
140
last_cyc
state_0
internal_wait
debugger_active
global_debugger_en
single_step_en
write_stack_ptr
(next_stack_ptr>sbk) AND (s[1:0]=11b)
(next_stack_ptr=sbk) AND (s[1:0]=01b)
(next_stack_ptr<sbk) AND (s[1:0]<10b)
opcode_fetch_adr_valid
opcode_fetch
single_step_trap
stack_trap
stack_trap_d1
stack_trap_d2
stack_trap_d3
D
Q
ENZ
CLK
prog_mux_control
CLK

FIG. 10

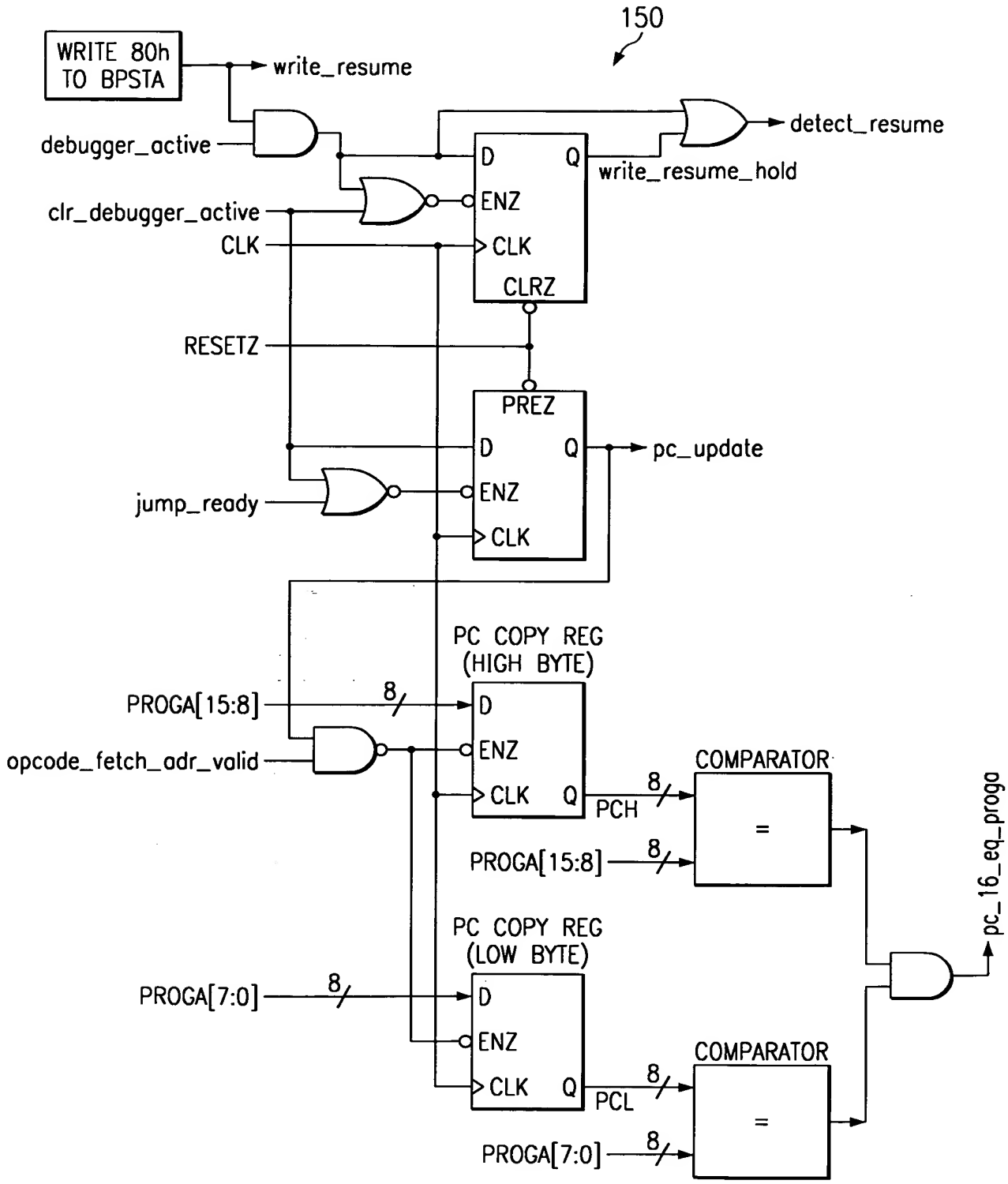
150
WRITE 80h TO BPSTA
write_resume
debugger_active
clr_debugger_active
CLK
RESETZ
jump_ready
D
Q
ENZ
CLK
CLRZ
PREZ
detect_resume
write_resume_hold
pc_update
PC COPY REG (HIGH BYTE)
PROGA[15:8]
8
opcode_fetch_adr_valid
PCH
COMPARATOR
=
PC COPY REG (LOW BYTE)
PROGA[7:0]
PCL
pc_16_eq_progo

FIG. 11

FIG. 12

160

DESCRIPTION	LABEL	ADDRESS
PORT-0	P0	80
STACK POINTER	SP	81
DATA POINTER LB	DPL	82
DATA POINTER HB	DPH	83
POWER CONTROL REG	PCON	87
TIMER/COUNTER CONTROL	TCON	88
TIMER/COUNTER MODE	TMOD	89
TIMER/COUNTER-0 LB	TL0	8A
TIMER/COUNTER-1 LB	TL1	8B
TIMER/COUNTER-0 HB	TH0	8C
TIMER/COUNTER-1 HB	TH1	8D
PORT-1	P1	90
SERIAL CONTROL REGISTER	SCOM	98
SERIAL DATA BUFFER	SBUF	99
PORT-2	P2	A0
INTERRUPT ENABLE REGISTER	IE	A8
PORT-3	P3	B0
INTERRUPT PRIORITY REGISTER	IP	B8
BPSTA: BREAK POINT STATUS REGISTER	BPSTA	BD
BPL1: BREAK POINT REGISTER-1 (LB)	BPL1	BE
BPH1: BREAK POINT REGISTER-1 (HB)	PBH1	BF
BNK1: BREAK POINT BANK REGISTER-1	BNK1	C0
BPL2: BREAK POINT REGISTER-2 (LB)	BPL2	C1
BPH2: BREAK POINT REGISTER-2 (HB)	PBH2	C2
BNK2: BREAK POINT BANK REGISTER-2	BNK2	C3
BPL3: BREAK POINT REGISTER-3 (LB)	BPL3	C4
BPH3: BREAK POINT REGISTER-3 (HB)	PBH3	C5
BNK3: BREAK POINT BANK REGISTER-3	BNK3	C6
BPL4: BREAK POINT REGISTER-4 (LB)	BPL4	C7
BPH4: BREAK POINT REGISTER-4 (HB)	PBH4	C8
BNK4: BREAK POINT BANK REGISTER-4	BNK4	C9
JTML: JUMP TO MONITOR ADDRESS REGISTER (LB)	JTML	CA
JTMH: JUMP TO MONITOR ADDRESS REGISTER (HB)	JTMH	CB
RESERVED		CC
RESERVED		CD
SBK: STACK BREAK POINT REGISTER	SBK	CE
BPCRL: BREAK POINT CONTROL REGISTER	BPCRL	CF
PROGRAM STATUS WORD	PSW	D0
D1→DF IS USED FOR SCRATCH PAD		D1→DF
ACCUMULATOR	A	E0
INTERRUPT ENABLE REGISTER-1	IE1	E8
B REGISTER	B	F0
RTKTM: RTK TIMER REGISTER	RTKT	F6
VECINT: VECTOR INTERRUPT REGISTER	VEC1	F7
INTERRUPT PRIORITY REGISTER-1	IP1	F8
PCL: PC COPY REGISTER (LB)	PCL	F9
PCH: PC COPY REGISTER (HB)	PCH	FA
WDCSR: WATCHDOG TIMER CONTROL AND STATUS REGISTER	WDCR	FB
MCNFG: MCU CONFIGURATION REGISTER	MCNFG	FC
WSGEN: WAIT-STATE GENERATOR REGISTER	WSGEN	FD
DSOVL: DATA-SPACE AND OVERLAY DEFINITION REGISTER	OVLAY	FE
BANK: BANK SELECT REGISTER	BANK	FF

162

164

166

7	6	5	4	3	2	1	0
A7	A6	A5	A4	A3	A2	A1	A0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7-0	A[7:0]	00h	SET-1: LOW-BYTE OF BREAK POINT ADDRESS

170

7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	A9	A8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7-0	A[15:8]	00h	SET-1: HIGH-BYTE OF BREAK POINT ADDRESS

172

7	6	5	4	3	2	1	0
BEN	RSV	RSV	RSV	B3	B2	B1	B0
R/W	R/O	R/O	R/O	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
3-0	B[3:0]	0h	SET-1 OF BANK, BREAK POINT ADDRESS
6-4	RSV	0h	RESERVED=0
7	BEN	0	BANK, BREAK POINT ENABLE/DISABLE BIT BEN=0 BANK, BREAK POINT ADDRESS IS DISABLED BEN=1 BANK, BREAK POINT ADDRESS IS ENABLED

174

FIG. 13

7	6	5	4	3	2	1	0
A7	A6	A5	A4	A3	A2	A1	A0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7-0	A[7:0]	00h	LOW-BYTE OF THE JUMP TO MONITOR ADDRESS

180

7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	A9	A8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7-0	A[15:8]	00h	HIGH-BYTE OF THE JUMP TO MONITOR ADDRESS

182

FIG. 14

7	6	5	4	3	2	1	0
A7	A6	A5	A4	A3	A2	A1	A0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION
7-0	A[7:0]	00h	STACK ADDRESS, USED TO COMPARE AGAINST THE STACK. IF A TRAP CONDITION IS DETECTED A LJMP TO MONITOR WILL BE INSERTED.

190

FIG. 15

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
R/0	R/0	R/0	R/0	R/0	R/0	R/0	R/0

BIT	NAME	RESET	FUNCTION
7-0	P[7:0]	00h	LOW BYTE OF THE PC. THIS VALUE IS LATCHED BY THE BREAK POINT LOGIC AND CAN BE READ ONLY BY MCU. MONITOR WILL USE THIS ADDRESS TO RESUME THE APPLICATION.

220

7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8
R/0	R/0	R/0	R/0	R/0	R/0	R/0	R/0

BIT	NAME	RESET	FUNCTION
7-0	P[7:0]	00h	HIGH BYTE OF THE PC. THIS VALUE IS LATCHED BY THE BREAK POINT LOGIC AND CAN BE READ ONLY BY MCU. MONITOR WILL USE THIS ADDRESS TO RESUME THE APPLICATION.

222

FIG. 18

200

7	6	5	4	3	2	1	0
BPE	STE	S1	S0	BE4	BE3	BE2	BE1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

BIT	NAME	RESET	FUNCTION	
0	BE1	0	ADDRESS BREAK POINT-1 CONTROL BIT	
			BE1=0	ADDRESS BREAK POINT-1 IS DISABLED
			BE1=1	ADDRESS BREAK POINT-1 IS ENABLED. IF A MATCH IS DECODED, THE JUMP LOGIC WILL BE TRIGGERED.
1	BE2	0	ADDRESS BREAK POINT-2 CONTROL BIT	
			BE2=0	ADDRESS BREAK POINT-2 IS DISABLED
			BE2=1	ADDRESS BREAK POINT-2 IS ENABLED. IF A MATCH IS DECODED, THE JUMP LOGIC WILL BE TRIGGERED.
2	BE3	0	ADDRESS BREAK POINT-3 CONTROL BIT	
			BE3=0	ADDRESS BREAK POINT-3 IS DISABLED
			BE3=1	ADDRESS BREAK POINT-3 IS ENABLED. IF A MATCH IS DECODED, THE JUMP LOGIC WILL BE TRIGGERED.
3	BE4	0	ADDRESS BREAK POINT-4 CONTROL BIT	
			BE4=0	ADDRESS BREAK POINT-4 IS DISABLED
			BE4=1	ADDRESS BREAK POINT-4 IS ENABLED. IF A MATCH IS DECODED, THE JUMP LOGIC WILL BE TRIGGERED.
5-4	S[1:0]	00b	STACK TRAP CONDITION 00b=NO STACK TRAP (STACK TRAP IS DISABLED) 01b=STACK TRAP ON SP=SBK 10b=STACK TRAP ON SP<SBK 11b=STACK TRAP ON SP>SBK	
6	STE	0	SINGLE STEP ENABLE/DISABLE CONTROL BIT. SEE SINGLE STEP FOR MORE EXPLANATION. STE=0 SINGLE STEP IS DISABLE STE=1 SINGLE STEP IS ENABLED	
7	BPE	0	GLOBAL DEBUGGER ENABLE/DISABLE CONTROL BIT	
			BPE=0	THE DEBUGGER LOGIC IS DISABLED. NO BREAK CAN HAPPEN. HOWEVER WRITING TO ALL DEBUGGER REGISTERS IS POSSIBLE.
			BPE=1	THE DEBUGGER LOGIC IS ENABLED

202 204 206 208

FIG. 16

7	6	5	4	3	2	1	0
RES	EA	SSP	SB	B4	B3	B2	B1
W/0	R/0	R/0	R/0	R/0	R/0	R/0	R/0

FIG. 17

BIT	NAME	RESET	FUNCTION	
0	B1	0	ADDRESS BREAK POINT-1 STATUS BIT	
			B1=0	ADDRESS BREAK POINT-1 DIDN'T CAUSED A BREAK CONDITION
			B1=1	INDICATES THAT ADDRESS BREAK POINT-1 CAUSED THE BREAK CONDITION. THIS BIT WILL BE CLEARED WHEN MCU WRITE "80h" TO THIS REGISTER.
1	B2	0	ADDRESS BREAK POINT-2 STATUS BIT	
			B2=0	ADDRESS BREAK POINT-2 DIDN'T CAUSED A BREAK CONDITION
			B2=1	ADDRESS BREAK POINT-2 CAUSED THE BREAK CONDITION. THIS BIT WILL BE CLEARED WHEN MCU WRITE "80h" TO THIS REGISTER.
2	B3	0	ADDRESS BREAK POINT-3 STATUS BIT	
			B3=0	ADDRESS BREAK POINT-3 DIDN'T CAUSED A BREAK CONDITION
			B3=1	ADDRESS BREAK POINT-3 CAUSED THE BREAK CONDITION. THIS BIT WILL BE CLEARED WHEN MCU WRITE "80h" TO THIS REGISTER.
3	B4	0	ADDRESS BREAK POINT-4 STATUS BIT	
			B4=0	ADDRESS BREAK POINT-4 DIDN'T CAUSED A BREAK CONDITION
			B4=1	ADDRESS BREAK POINT-4 CAUSED THE BREAK CONDITION. THIS BIT WILL BE CLEARED WHEN MCU WRITE "80h" TO THIS REGISTER.
4	SB	0	STACK TRAP STATUS BIT	
			SB=0	STACK TRAP DIDN'T CAUSED A BREAK CONDITION
			SB=1	STACK TRAP CAUSED THE BREAK CONDITION. THIS BIT WILL BE CLEARED WHEN MCU WRITE "80h" TO THIS REGISTER.
5	SSP	0	SINGLE STEP BREAK POINT STATUS BIT	
			SSP=0	SINGLE STEP BREAK POINT DIDN'T CAUSED A BREAK CONDITION
			SSP=1	SINGLE STEP BREAK POINT CAUSED THE BREAK CONDITION. THIS BIT WILL BE CLEARED WHEN MCU WRITE "80h" TO THIS REGISTER.
6	EA	0	REFLECTS THE REAL VALUE OF EA BIT WHEN IN DEBUG MODE. SEE SINGLE STEP FOR MORE EXPLANATION. EA=0 INTERRUPT IS DISABLED EA=1 INTERRUPT IS ENABLED	
7	RES	0	RESUME CONTROL BIT. WRITING A "80h" TO THIS REGISTER WILL WRITE-PROTECT ESFR(BE-CF), ENABLE THE PCL/PCH UPDATE AND CLEAR B[4:1] BITS. THIS BIT IS READ AS "0". WRITING A "55h" TO THIS REGISTER WILL UNPROTECT ESFR[BE-CF] BUT NOT CLEAR THE STATUS BITS.	

210

212

214

216

218